Microkernels

With a monolithic kernel such as the Linux kernel, memory is divided into user space and kernel space. Kernel space is where the actual kernel code is loaded, and where memory is allocated for kernel-level operations. Kernel operations include scheduling, process management, signaling, device I/O, paging, and swapping: the core operations that other programs rely on to be taken care of. Because the kernel code includes low-level interaction with the hardware, monolithic kernels appear to be specific to a particular architecture.
A microkernel performs a much smaller set of operations, and in more limited form: interprocess communication, limited process management and scheduling, and some low-level I/O. Microkernels appear to be less hardware-specific because many of the system specifics are pushed into user space. A microkernel architecture is basically a way of abstracting the details of process control, memory allocation, and resource allocation so that a port to another chipset would require minimal changes.
Typical Microkernels provide only most essential functions as part of the
Kernel

What functions?
- Process Management
- Inter-process Communication (IPC)
- Memory Management
- Device I/O
- and others. . .

Typical Microkernels (continued)
_ User-level processes:
- Device Drivers
- Networking
- File Systems
- VM Paging

_ Strengths:
- *Portability* : Small size, HLL
- *Flexibility* : User-level OS functions

_ Weaknesses:
- *Performance*: Increased communication across kernel boundaries

What Is A Microkernel? 1

The traditional operating system is built within a monolithic kernel. This kernel contains all of the core functions of the operating system, from scheduling to the management of the file system. In contrast microkernel systems are essentially monolithic kernels that have been cut down to the bear minimum. Microkernel systems only provide the essential system functions. Any other operating system services are provided outside the kernel by user processes (called servers). As advances are made in computing it is difficult to see the distinction between monolithic kernels and microkernels. For example monolithic kernel based operating systems such as Windows 95 have threads, which is an inherent feature of microkernel architectures.

The many companies currently involved in the development of microkernels cannot agree on what services the microkernel should provide. For example should the device drivers be situated within the kernel? Mach and Chorus keep the device drivers outside the kernel, but Windows NT allows them to run in kernel mode to improve efficiency. Running these services in kernel mode means message passing is replaced by function calls, which are a lot more efficient. Should non-kernel operations be run in user space, is another issue? Under NT many servers run their code in kernel space for reasons of efficiency. This has led to many companies stating that NT is not a true microkernel.

Typically the following functions are handled within the microkernel, although as explained later this is not a definitive list:

- Inter-process Communication (via message passing)
- Short-term scheduling (basic thread process management)
- Very low-level memory management
- Low level Input/Output (mostly interrupt handling)
- Low level Network support

The main goal of a microkernel system is to keep it small. The services contained within the kernel itself are only provided because it would be expensive, or difficult to provide them elsewhere. The microkernel is the layer between the physical hardware, and the set of sub-systems that comprise the remainder of the Operating System. The user-level servers that provide the other operating system functions run outside the kernel in what is called *user mode*. Any processes within the kernel are said to run in *kernel mode*. The user level processes that comprise the sub-system typically provide the following services:

- The File system
- Directory system
- Full process management (scheduling)
- Security services

The microkernel replaces the vertical layout of the monolithic operating system and replaces it by a horizontal one. The microkernel facilitates communication between the components of the operating system. The kernel validates messages and either deals with them itself or passes them between servers. The user level services are granted access to hardware resources by issuing requests to the kernel via function calls. Although communication to hardware is done low level through the kernel, this is transparent to the user. The user simply calls a function through the server and the kernel and server communicate to accomplish the task. The microkernel itself is never scheduled for execution; its simply called by processes or hardware interrupts.